Visual Odometry with Kalman Filter

Anisha Singrodia, Renu Reddy Kasala
{singroa, renu1}@seas.upenn.edu

I. Abstract

In this project, the egomotion of the vehicle is estimated from a sequence of stereo images using Visual Odometry (VO). Since Visual Odometry is used to calculate the movement of the camera through continuously adding new poses, any inaccuracies in each successive frame-to-frame motion will build up over time. As a result, the estimated trajectory gradually deviates from the actual path, creating drift. With the goal of reducing the error drift, implemented an Kalman Filter (KF) to get better state estimates. The algorithm's effectiveness is showcased through experiments on KITTI Vission Benchmark Suite[1]. These experiments provide sufficient evidence of the algorithm's capabilities, and demonstrate its ability to achieve results on a range of tasks.

II. Introduction

As autonomous vehicle gains more relevance in practical applications, it has become evident that the use of cameras for navigation is essential. Hence, Visual odometry has become a crucial task for intelligent systems to tackle while interacting with their dynamic environment. One main hurdle of using such techniques in real world is the noise present in the data acquired through not so perfect sensors. Consequently, filtering techniques are necessary to eliminate sensor noise efficiently and enable robust estimation. In this project we implemented a Kalman Filter on Stereo Visual Odometry outputs. Then tested the robustness and effect of this filter in real world scenarios using KITTI dataset. The final estimated results are tabulated. Further, we discussed the conclusions and improvements to the system.

A. Contributions

(i) Comparison of ORB, SIFT and FAST Feature detectors.
(ii) Implementation of Stereo Visual Odometry using KITTI dataset.
(iii) Implementation of Kalman filtering on the VO outputs.
(iv) Comparing the filtered estimates and VO outputs with the ground truth.

III. Background

VO is defined as the process of estimating the robot's motion (translation and rotation with respect to a reference frame) by observing a sequence of images of its environment [2]. A Kalman filter is an optimal estimator - infers parameters of interest from indirect, inaccurate and uncertain observations. It is recursive so that new measurements can be processed as they arrive. The recursion can be broken up into two steps, namely, a propagation step and an update step. The propagation step projects the current state of the system forward by using a process model of the system with some added noise. The update step then corrects this propagated step by incorporating measurement data using a sensor measurement model [3].

IV. Related Work

There are many publications on visual odometry systems. One of the most popular implementations is the ORB-SLAM3 [4] implementation. This is considered a state-of-the art SLAM system that is able to perform visual, visual-inertial, and multi-map SLAM with a number of different sensors and camera models. Though this is a SLAM system, the boundary between SLAM systems and VO systems is not clearly defined, especially in VO systems that implement loop-closing. Another popular VO system is VINS-Mono [5]. This system uses a single camera and an inertial measurement unit to perform six degrees-of-freedom state estimation. It makes use of pre-integrated IMU measurements and feature observations to determine this pose.

V. Approach

We took following steps to develop our Visual Odometry model and validate its performance:

A. KITTI Dataset Overview

KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) is one of the popular datasets used in autonomous navigation. It consists of the following:

- Sequences: where each sequence contains the left and right images.
- Calibration data: for the cameras: P0 and P1 are the 3x4 projection matrices after rectification. Here P0 denotes the left and P1 denotes the right camera.
- Timestamps: for each of the synchronized image pairs in seconds
- Poses: contains the ground truth poses of the trajectory. Each file contains a N x 12 table, where N is the number of frames of this sequence. Row i represents the i^th pose of the left camera coordinate system (i.e., z pointing forwards) through a 3x4 transformation matrix.

B. Overview of Stereo Visual Odometry

For Stereo VO, two images (left and right) are needed for each of the consecutive steps. To compensate for lens distortion, the images are then processed. All the steps in the Stereo VO pipeline are discussed below.

1) Feature Detection and Matching: Feature detection is crucial to identify the key features in an image that help in tracking the optical flow and how the camera is moved between consecutive images, or how the scenes changes with time. To identify and match the features, stereo rectification is done so that epipolar lines become parallel to horizontal. This way we can find the corresponding features by searching along the corresponding horizontal lines. In KITTI dataset the input images are already stereo rectified and are corrected for lens distortion.

We tried the following Feature Detectors:

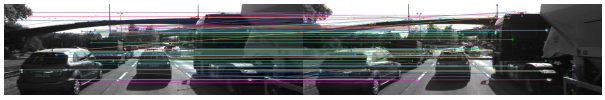
- Feature Detection using FAST - Features from Accelerated Segment Test algorithm: FAST is a corner detection algorithm which identifies corners, i.e., key points in an image that have high gradient values in very fast and efficient manner.
- Feature Detection using ORB - Oriented FAST and Rotated BRIEF: is the combination of two algorithms FAST and BRIEF.
- Feature Detection using SIFT - Scale Invariant Feature Transform: The keypoints obtained are scale rotation invariants.

(a) Sift Feature Detection

(b) ORB Feature Detection

(c) FAST Feature Detection

Fig. 1: Feature Detection Comparision (500 features)



(a) Sift Feature Matching

(b) ORB Feature Matching

(c) FAST Feature Detection

Fig. 2: Feature Matching Comparison (100 features matching)

These three were specifically chosen since they are widely used for feature detection. The performance was tested based on the speed and the percentage of features matched. Due to our limited computational capacity we chose to use FAST detectors over ORB and SIFT despite ORB detector giving better accuracy. Since speed was our main concern and we tolerated some loss in accuracy.

Generated a disparity map using cv2.StereoSGBM_create(). The disparity map refers to the motion between a pair of stereo images. This generated disparity map is used to do feature matching using FAST detector. Here, we haven't directly applied the feature matching algorithm to the Left and Right images cause it usually tend to generates features that are usually concentrated in one edge of the image. Instead, divided each image into tiles and found ten important features in each of the tiles from left and right images.

But from the above images it is clear that best features obtained from FAST algorithm are more concentrated at one region in the whole image. So in order to get good features distributed over whole image space we divided image into smaller tiles and run FAST over each of them. This way we were able to get good features spread across the image with lower computational time.

2) Feature Tracking: To know how the points have moved in the image from one frame to the next, we used Optical Flow. Optical flow refers to the apparent movement pattern of objects in an image between two successive frames, which can be caused by either the motion of the camera or the objects themselves. It is represented as a 2D vector field, where each vector represents the displacement of individual points from the first frame to the second frame. We set a threshold value to check for the flow. Optical Flow assumptions include: Brightness Constancy, Locally Uniform Motion, Small Motions, No Occlusions and Invertibility.

3) Triangulation of Feature Points: Using the consecutive images and key features, we determined world coordinates of left and right image frames using a process called triangulation. We first identified key features in two left images of consecutive frames. Using disparity we found corresponding features in both right images of the consecutive frames. Now using triangulation we projected these features in world frame and obtained 3D points as seen from both the frames. Now, we reprojected these 3d points from two frames to obtain pose in form of rotation matrix and translation using least squares optimization.

4) Calculating Pose from VO: The motion of the camera between two consecutive frames is estimated by minimizing the error in re-projecting the image for all corresponding feature points. This involves finding the corresponding world coordinates for the matching feature points at two consecutive time steps. Re-projection of the world coordinates back into the image using a transformation is done in order to minimise the difference between the true and projected points.

C. Kalman Filter

To further improve the trajectory obtained from Visual Odometry, we implement a kalman filter. We aim to decrease the error between VO and Ground truth trajectories. We don't have dynamics defined for the system so we modeled a gaussian noise as our dynamics. We have taken our state as x, y coordinates of the position and θ as the yaw of the pose of body.

$$X = [x, y, \theta]$$

Implementation of Kalman Filter involves following steps:

- Propagation step: In this step, we add a gaussian noise ϵ with mean 0 and a fixed standard deviation. We iterated through various different values of standard deviation and found the best value among them.

$$x_{k+1|k} = x_{k|k} + \epsilon$$

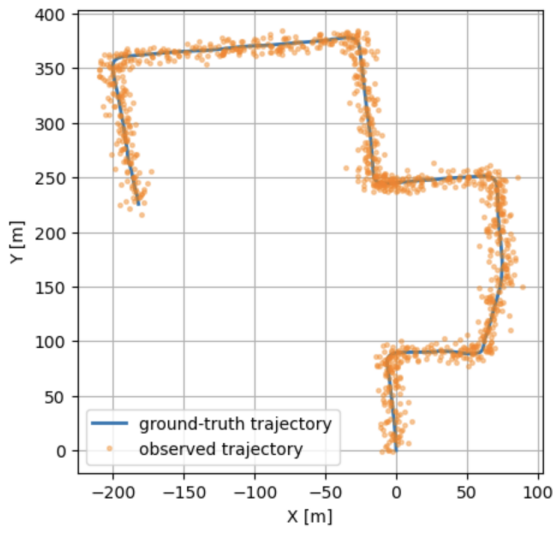


Fig. 3: Addition of gaussian noise in VO output

- Update step: In this step, we use observations from our Visual Odometry for each time step to update the trajectory. We first calculate Kalman gain for our filter using Jacobian H of our observation function, covariance matrix of error from previous step and observation noise covariance matrix Q.

$$K = PH^T(HPH^T + Q)^{-1}$$

We then update our state using this Kalman gain and observation y as follows:

$$x_{k+1|k+1} = x_{k+1|k} + K\delta$$

where δ is difference between observed and previous state. Iterating the two steps over the all time steps gives us a better estimate of trajectory.

Algorithm 1 Kalman Filter

$\mu_0 \leftarrow [x_0, y_0, \theta_0]$
$Cov_0 \leftarrow P$
for $t = 1, 2, \ldots, timesteps$ **do**
    Propagate states
    $x_{t+1|t} \leftarrow x_{t|t} + \epsilon$
    Calculate Kalman Gain
    $K \leftarrow PH^T(HPH^T + Q)^{-1}$
    Update states using observations
    $x_{t+1|t+1} \leftarrow x_{t+1|t} + K\delta$

VI. Experimental Results

A. Comparison with Ground Truth

We analysed the performance of our project by primarily plotting and calculating the error between the ground truth and the estimate. This is the most intuitive and a standard way of measuring performance.

Performance Metric	**VO** (in meters)	**VO+KF** (in meters)
Mean Error	7.5863	6.0266
Median of Error	7.7652	5.7736
Error Standard Deviation	2.5712	2.7558
Root Mean Square Error	8.0102	6.6268
Minimum Error	0.0	0.0
Maximum Error	10.8039	12.7537

TABLE I: Performance Statistics for the visual odometry

1) Overall Trajectory in 2D: Below is the 2D trajectory obtained from ground truth pose, Visual odometry and Kalman filtering. We obtained ground truth poses from poses.txt file provided by KITTI dataset.

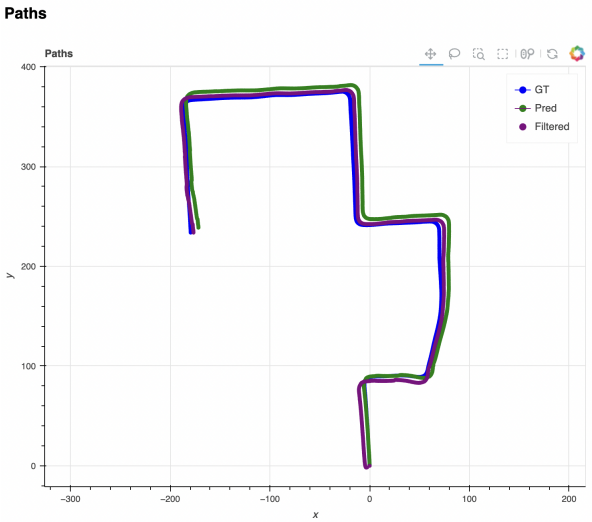


Fig. 4: VO path Vs Filtered path

2) Error in Translation: To measure the error in trajectory we plotted the estimated x,y translation, VO translation against the ground truth as shown below:

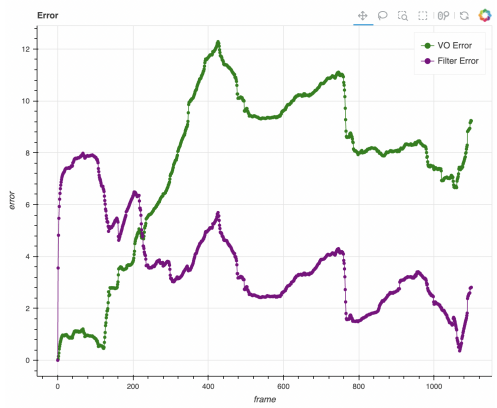
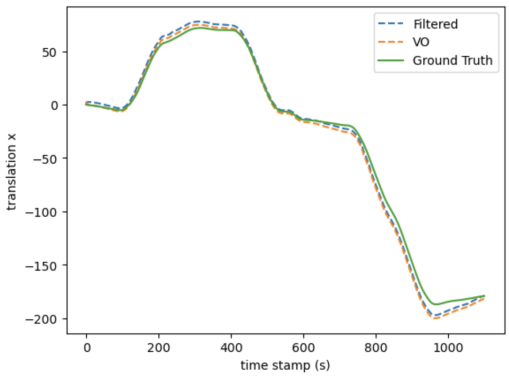
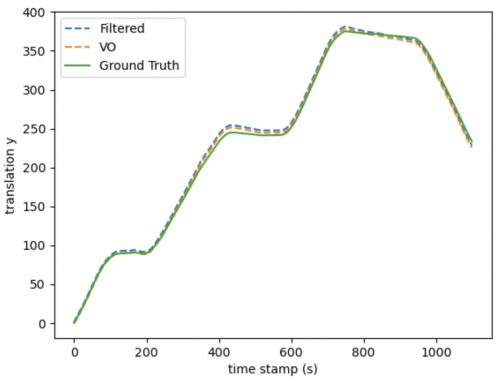


Fig. 5: VO path Error Vs Filtered path Error with respect to ground truth



(a) Translation along x direction



(b) Translation along y direction

Fig. 6: Filter Vs Ground truth Vs VO Translation

VII. Discussion

Through experimentation with KITTI Dataset, we concluded that the pose estimation can be improved using Kalman filter over VO when compared to basic Stereo VO implementation. Although, the predicted estimates are good, they are not sufficient to be used for real life applications. Due to restrictions in computational capacity, we used FAST feature detectors, and Greyscale images. Using better feature detectors and color images will further improve the results. Since we did not know the dynamics, we have used gaussian noise as our dynamics, but having IMU data along with Stereo images would've enabled us to use the linear and angular velocities to implement an Extended Kalman Filter instead of a regular Kalman filter. Further testing on more data will give better insights to apply it to real world scenarios.

References

[1] Geiger2012CVPR, Andreas Geiger and Philip Lenz and Raquel Urtasun, Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite, Conference on Computer Vision and Pattern Recognition (CVPR), 2012

[2] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics". In: Intelligent Industrial Systems 1 (Nov. 2015). DOI: 10.1007/ s40903-015-0032-7.

[3] Lindsay Kleeman, Department of Electrical and Computer Systems Engineering, CMU, "kleeman understanding kalman.pdf", sbp papers.

[4] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," in IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1874-1890, Dec. 2021, doi: 10.1109/TRO.2021.3075644.

[5] Davide Scaramuzza and Friedrich Fraundorfer. "Visual Odometry [Tutorial]". In: IEEE Robotics Automation Magazine 18.4 (2011), pp. 80–92. DOI: 10.1109/MRA.2011.943233.

[6] D. Nister, O. Naroditsky, and J. Bergen. "Visual odometry". In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. Vol. 1. 2004, pp. I–I. DOI: 10.1109/CVPR.2004.1315094.